Uniwersytet Jagielloński Instytut Informatyki KIS Katedra Informatyki i Stosowanej	Metody programowania 2023/2024 Kolejki priorytetowe na drzewach BST	P_09

Opis:

Napisz efektywną aplikację działającą na kolejce priorytetowej MIN/MAX zawierającej elementy typu *Person*, w których priorytety się nie powtarzają i zrealizowanej jako drzewo *BST*. Przy czym węzły drzewa *BST* mają postać:

```
class Node {
   public Person info; // element danych (klucz)
   public Node left; // lewy potomek węzła
   public Node right; // prawy lewy potomek węzła
      …
}   // koniec klasy Node
```

natomiast klasa *Person* ma postać:

```
class Person {
     public int priority; // priorytet osoby
     public String name;
     public String surname;
       …
}  // koniec klasy Person
```

Aplikacja powinna zawierać się trzy moduły: edycji, kolejkowania oraz raportowania.

Moduły przedstawiają się następująco:

- **Moduł edycji**: powinien zawierać operacje:
  - *CREATE order n* $p_1 \ldots p_n$ – *rekurencyjnie* tworzy drzewo BST na podstawie listy *n* elementów p_1 do p_n *typu Person,* podanych w porządku ich priorytetów, wyznaczonym przez argument *order* (*PREORDER* albo *POSTORDER*).
    Przy czym jeśli drzewo już istnieje, powinno zostać zastąpione nowym.
  - *DELETE x* – usuwa z drzewa osobę o zadanym priorytecie x . W przypadku, gdy w drzewie usuwany węzeł ma dwóch potomków, zamienia go z jego *następnikiem*, przy czym jeśli osoba o priorytecie x nie występuje w kolejce to operacja wypisze informację: DELETE x: BRAK.

- **Moduł kolejkowania**: powinien implementować podstawowe operacje na kolejce:
  - *ENQUE p* – dodaje element *p* typu *Person* do kolejki.
  - *DEQUEMAX* – usuwa osobę o najwyższym priorytecie.
  - *DEQUEMIN* – usuwa osobę o *najniższym* priorytecie,
    przy czym obie operacje wypiszą DEQUEMAX: lub DEQUEMIN: a po nich dane o usuwanej osobie, jak w poniższym przykładzie.
  - *NEXT* x – zwraca najbliższą osobę o *priorytecie większym* od x.
  - *PREV* x – zwraca najbliższą osobą o priorytecie mniejszym od x,
    przy czym jeśli osoba o priorytecie x nie występuje w kolejce lub gdy nie ma w kolejce najbliższych osób o priorytecie mniejszym lub większym od x, obie powyższe operacje wypiszą NEXT x: BRAK lub PREV x: BRAK

- **Moduł raportowania**: powinien realizować następujące komendy:
  - *PREORDER* – wypisuje listę osób w porządku preorder.
  - *INORDER* – wypisuje listę osób w porządku inorder.
  - *POSTORDER* – wypisuje listę osób w porządku postorder.
  - *HEIGHT* – *rekurencyjnie* zwraca wysokość drzewa

Wejście:

Dane do programu wczytywane są ze standardowego wejścia (klawiatury), zgodnie z poniższą specyfikacją:

Pierwsza linia zawiera liczbę całkowitą $(1 \leq z \leq 100)$ oznaczającą ilość testów.

- Pierwsza linia testu zawiera liczbę całkowitą $(1 \leq k \leq 100)$ oznaczającą ilość komend do wykonania na kolejce.
- W każdej następnej linii znajduje się jedna z wymienionych wyżej operacji i ewentualnie jej argument(y).
- W operacjach *CREATE* oraz *ENQUE* jako argumenty przyjmuje się trójkę: (**`priorytet, imię, nazwisko).`** W operacjach *DELETE*, *PREV* oraz *NEXT* podawany jest tylko argument **priorytet**.

Wyjście:

Każdy test powinien zaczynać się od wypisania ciągu „ZESTAW n” (od 1). Dla każdej operacji w zestawie wypisz w jednej linii jej wynik zgodnie z podanymi przykładami.

Wymagania implementacyjne:

1. Jedynym możliwym importem jest skaner wczytywania z klawiatury.
2. Wszystkie komendy oprócz *CREATE* oraz *HEIGHT* muszą być zaimplementowane w wersji iteracyjnej.
3. Można założyć że"
   (a) żadna operacja nie zostanie wywołana na kolejce przed wykonaniem operacji CREATE oraz
   (b) żadna operacja nie zostanie wywołana dla kolejki pustej.
4. Złożoność wszystkich operacji powinna być optymalna. W szczególności, żadna operacja nie powinna wykonywać więcej niż jednej pętli od korzenia drzewa np. operacje NEXT x lub PREV x znajdują osobę o priorytecie x, następnie szukają następnika lub poprzednika x nie korzystając z operacji search(x) i parent(x).
5. Operacje *PREV* x oraz *NEXT* x nie mogą korzystać z listy INORDER
6. Przypominam o komentowaniu aplikacji w formie opisanej w punkcie 3 Regulaminu zaliczania programów na BaCy z roku 2023/2024.

Przykładowy test:

test.in:

```
1
19
CREATE POSTORDER 4 12 Adam Nowak 33 Marek Mickiewicz 43 Zofia Krzak 37 Ola Nowicka
POSTORDER
PREORDER
INORDER
PREV 12
NEXT 43
NEXT 33
PREV 33
PREV 50
NEXT 50
DEQUEMAX
DEQUEMIN
INORDER
DELETE 50
```

DELETE 33
POSTORDER
ENQUE 35 Andrzej Wolny
INORDER
HEIGHT

test.out:

ZESTAW 1
POSTORDER: 12 - Adam Nowak, 33 - Marek Mickiewicz, 43 - Zofia Krzak, 37 - Ola Nowicka
PREORDER: 37 - Ola Nowicka, 33 - Marek Mickiewicz, 12 - Adam Nowak, 43 - Zofia Krzak
INORDER: 12 - Adam Nowak, 33 - Marek Mickiewicz, 37 - Ola Nowicka, 43 - Zofia Krzak
PREV 12: BRAK
NEXT 43: BRAK
NEXT 33: 37 - Ola Nowicka
PREV 33: 12 - Adam Nowak
PREV 50: BRAK
NEXT 50: BRAK
DEQUEMAX: 43 - Zofia Krzak
DEQUEMIN: 12 - Adam Nowak
INORDER: 33 - Marek Mickiewicz, 37 - Ola Nowicka
DELETE 50: BRAK
POSTORDER: 37 - Ola Nowicka
INORDER: 35 - Andrzej Wolny, 37 - Ola Nowicka
HEIGHT: 1